AAU-Star and AAU Honeyjar

Malware Analysis Platforms Developed by Students

Pedersen, Jens Myrup; Stevanovic, Matija

Published in:
Image Processing and Communications Challenges 7

DOI (link to publication from Publisher):
10.1007/978-3-319-23814-2_32

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

AAU-Star and AAU Honeyjar: Malware Analysis Platforms Developed by Students

Jens Myrup Pedersen and Matija Stevanovic

Abstract In this paper we will demonstrate how systems for malware testing can be designed, implemented and used by master degree students. In this way, we have established two strong platforms for malware testing, while at the same time provided the students with a strong theoretical and practical understanding of how to execute, analyse, and classify malware based on their network and host activities.

1 Introduction

Malware is becoming an increasing problem in computers and computer systems, with severe economic consequences for individuals, businesses, and societies [8]. On the individual basis, computers can be infected with e.g. spyware that is used for stealing personal information or ransomware, which destroy all date on the infected machine unless a ransom is paid. For businesses, infected machines are vulnerable to the same kind of attacks, but often with more severe consequences such as stealing business secrets or destruction of data that is crucial to the operation of the company. Another particular kind of malware, botnets, makes a strong tool for cyber criminals [6, 9]. Computers infected with bot malware (also known as zombies) can be remotely controlled and used for a number of malicious activities, often targeting business or infrastructure that is critical for society. These include Distributed Denial of Service attacks, where a service is left unavailable due to a huge number of requests coming from different machines. Recent years have seen attacks on e.g. Sony and Mastercard. In Denmark, the personal identification system used to access all public and many private services, was recently successfully attacked. Also other malicious activities such as Click Fraud, Pay Per Install and sending of SPAM emails are supported by botnets.

J.M. Pedersen (✉) · M. Stevanovic
Department of Electronic Systems, Aalborg University, Aalborg, Denmark
e-mail: jens@es.aau.dk

M. Stevanovic
e-mail: mst@es.aau.dk

R.S. Choraś (ed.), *Image Processing and Communications Challenges 7*,
Advances in Intelligent Systems and Computing 389,
DOI 10.1007/978-3-319-23814-2_32

Malware can be fought at different levels: One is of course protection at device level, using anti-virus systems. This is generally efficient, but suffers from zero day attacks (e.g. new malware which is not yet known by the systems) and the risk of users overruling the systems if manipulated through e.g. social engineering. Another approach, which is a research topic at Aalborg University, is to detect malware through the analysis of network traffic. The hypothesis here is that malicious traffic can be distinguished from benign traffic using e.g. machine learning algorithms. Several authors have proposed detection algorithms based on the these assumptions, such as Bilge et al. [2], Gu et al. [4], Strayer et al. [10], Zhao et al. [12]. However, building, improving, analyzing and testing such tools and algorithms is a data intensive task with quite a few challenges. First, we need to collect a large amount of data from many different malware samples, and it needs to be correctly labeled (as either malicious/non-malicious or with a division of the malware into types of malware). Second, it must be done without the risk of harming others, meaning that any test environment should have only limited access to the Internet (with no harmful traffic let through). Third, the computers infected should ideally "look like" normal non-monitored computers, used as a normal user would do, in order to avoid that the malware becomes suspicious of the environment which could trigger a different behavior.

This paper demonstrates how student projects can be used as a mean of building platforms and systems for malware testing, which fulfills these criteria to a large extend.

2 Background

Aalborg University has a long tradition for Problem Based Learning [7], where students spend approximately half of their time during each semester on problem-based project work. This is usually done in groups, in the master programme with 2–4 students per group. There is also some tradition for mega projects, where student groups from different educations and/or semesters develop sub-projects which are part of a larger project, which is often developed over several years. Examples of these are AAU Unicorn racer car [11] and the AAU Student satellites [1].

The first step towards AAU HoneyJar, which is also such a mega project, was taken in 2013, and has been followed up by additional projects in the following years. They are carried out within the master programme of Networks and Distributed Systems. The main motivation for developing the malware testing platforms is to provide an environment for reliable and secure testing of malicious software with the goal of obtaining both network and client-level behavioral forensics. Having a "quality" data sets is one of the most important prerequisite for developing malware detection approaches. Here under quality we mean a substantial amount of data that successfully captures malware activity.

Within the series of student projects that were realized in connection with AAU HoneyJar many of them have relied on machine learning algorithms (MLAs) for

identifying the patterns of network traffic produced by malware and the behavior of malware at client machines. As MLAs represent a class of data driven approaches they are dependent on the quality of the data sets used for both training and evaluation. The student projects were using data captured by the HoneyJar in order to develop malware detection approaches based on both supervised and unsupervised MLAs. The developed detection methods targeted malware on both client and host levels. At network level students used supervised machine learning to classify benign from botnet network traffic. At client level supervised MLAs were used to classify software as malicious and then to classify the malicious software to an appropriate malware family. In addition, one project analyzed if malware behavior can be clustered using unsupervised MLAs and if the clusters extracted this way correspond to the malware types defined by anti-virus providers.

3 The Systems Developed at AAU

This section describes the two main projects, which are run and developed by students at Aalborg University: AAU HoneyJar, which is an automatized and secure malware testing environment, and Star-AV, which is a virtualized malware testing development.

3.1 AAU HoneyJar: An Automatized and Secure Malware Testing Environment

AAU HoneyJar is the oldest of the two projects, and has been developed and used since 2013. The basic architecture is shown in Fig. 1. It consists of the following elements, which also makes it a natural division between different student projects:

- The Test Environment consists of a number of computers (inmates), which can be used to run malware samples. It should be automatized as much as possible, and in particular, it should be fast and easy to upload and run malware, to clean/swipe the computer after running a malware sample, and to configure each inmate as needed for the testing purposes. Moreover, it should be possible to have the inmates "behave" as if real users are interacting with them, e.g. by developing scripts to open webpages, login to email/Facebook accounts, perform Google searchers etc. The inmates can be virtual or physical machines, or a combination hereof. Another important part of the Test Environment is an emulated Internet environment, which makes the inmates believe that they are connected to the Internet. By having basic services, such as Windows Time and DNS, available from the emulated Internet, access to the real Internet can be limited. This automatized test setup was developed through a number of student projects during the previous years.

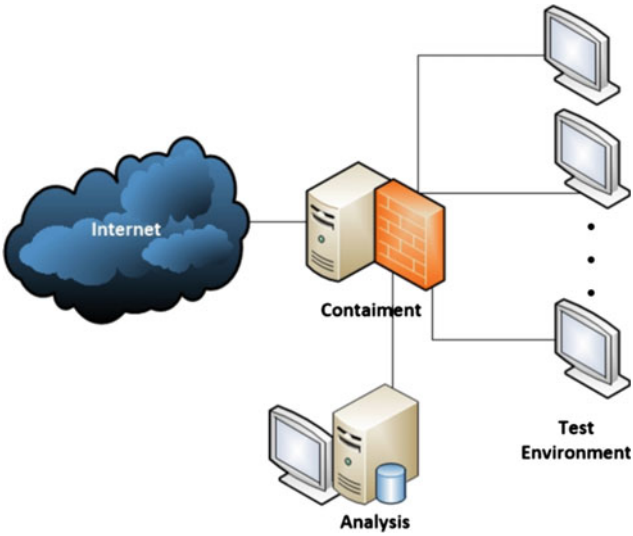

Fig. 1 Architecture of AAU Honey Jar

- The Containment part filters what traffic can enter the real Internet. Giving the infected inmates unfiltered and unlimited Internet access carries a large risk of being active part in DDoS attacks or other malicious activities, including the infection of other user's computers and stealing of personal information. As a university, this risk is unacceptable. On the other hand, with no access to the Internet it is not possible to study the communication between inmates and e.g. command and control servers, which cannot be emulated through the "emulated Internet". Our philosophy is to block all harmful traffic, but to allow traffic through that is considered harmless. This is not easy to define (and even more difficult to decide in real-time). Different strategies has been used in two different student projects, one being to initially block all traffic, but as traffic is observed and deemed harmless to whitelist what is considered harmless traffic. This was partially done in an automatized manner, by using TCP injection to let the inmate reveal the content of the first packet with payload.
- The analysis part is then used for actually analyzing the data and using them for e.g. training and testing of machine learning algorithms. This has been done not only in student projects, but also as a part of PhD research projects, demonstrating the value of AAU HoneyJar.

3.2 Star-AV: Virtualized Malware Testing Environment

The Star-AV project was initiated in 2014. While it is currently a separate project, it might over time be integrated into the "Test Environment" of AAU HoneyJar. The idea behind the Star-AV project is to create a scalable system, that is able to test

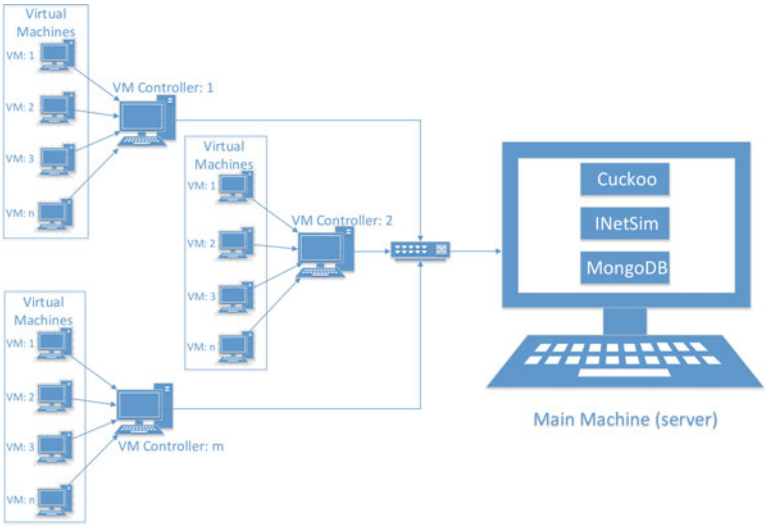

Fig. 2 Architecture of Star-AV

large amounts of malware in a highly automatized fashion. The students has been working on the project during two semesters (Fall 2014, and Spring 2015), and the fall semester part already resulted in a conference paper [5]. Moreover, the students received the Danish Telecommunication Price for Students 2015 based on this work.

Based on a modified version of the Cuckoo Sandbox [3], the system consists of a "master computers" along with a number of computers, where each computer is hosting a number of virtual machines. The basic architecture is shown in Fig. 2. Each machine runs the Microsoft Windows 7 Operating System, along with a number of installed programs including Skype and Adobe Reader. Moreover, each computer runs a script to emulate web activity, to decrease the probability that the malware will detect it is in a contained environment.

The reason for using Cuckoo Sandbox is that it allows for monitoring of the malware behavior on the host side, as opposed to study only network activity. This makes it possible to monitor e.g. accessed files, mutexes, registry keys and windows APIs in addition to network activity.

During the most recent project (a master thesis), the setup was used to test more than 300,000 malware samples, providing valuable data for training algorithms to distinguish between different malware families.

4 Future Work

While both platforms are already fully functional, and useful for both research and education, they also both have a good potential for further development.

For AAU HoneyJar, all of the three subprojects have the potential to be developed even further.

For the Test Setup, it could be particularly interesting to improve the emulation of human behavior as well as the emulated Internet. We believe that this could lead to more malware becoming active when we execute the malware. Moreover, increasing the number of virtual/physical machines would make it possible to run each malware sample for a longer duration of time, also increasing the probability that it will become active.

For the Containment, more sophisticated TCP injection could be an interesting project. In particular, it would be interesting if a conversation could be intercepted and "held in the air" until it was approved. This would require translation of ACK and SEQ numbers, but would be more difficult to detect than in the current implementation (once a connection is whitelisted, it must be reset before a handshake with the server on the Internet is established).

For the Analysis part, it could be interesting to explore the connection between what is measured on the computer using Star-AV, with network activities including analysis of both data and DNS traffic.

For the Star-AV, it could be considered if monitoring of cleanware could be improved. In particular, it would be interesting to see if not only execution, but also installation could be monitored. This is not possible today, due to the large amounts of data involved. Since we can today monitor both installation and execution of malware, we believe that this could improve the training data and thus the accuracy of the resulting classification algorithms.

5 Conclusion

In this paper, we have showed how master student projects can contribute to designing and implementing systems for malware testing. Two strong platforms have been presented, the AAU HoneyJar and Star-AV. These platforms make it possible to execute and monitor behavior of large amounts of malware in a secure setting, thus providing valuable data for education and research purposes. In particular, the projects have made it possible to train and test machine learning algorithms for malware detection. The projects reflect the tradition of Problem Based Learning at Aalborg University, and serves as good examples of mega projects. It is expected that both platforms will be further developed through future student projects.

References

1. AAU StudentSpace: Aalborg University Denmark. http://space.aau.dk (2015)
2. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC'12, pp. 129–138. ACM, New York (2012)
3. Cuckoo sandbox. http://cuckoosandbox.org (2015)

4. Gu, G., Perdisci, R., Zhang, J., Lee, W., et al.: Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection. USENIX Secur. Symp. **5**, 139–154 (2008)
5. Hansen, S.S., Larsen, T.M.T., Pirscoveanu, R.S., Czech, A., Stevanovic, M., Pedersen, J.M.: Analysis of malware behavior: systemtype classification using machine learning. In: Cyber Situational Awareness, Data Analytics and Assessment (CyberSA 2015). IEEE, C-MRiC (2015)
6. Hogben, G., Plohmann, D., Gerhards-Padilla, E., Leder, F.: Botnets: Detection, measurement, disinfection and defence. In: European Network and Information Security Agency (2011)
7. Kolmos, A., Krogh, L., Fink, F.K.: The Aalborg PBL model: progress, diversity and challenges. Aalborg University Press, Aalborg (2004)
8. Net losses: Estimating the global cost of cybercrime. McAfee, Centre for Strategic & International Studies (2014)
9. Silva, S.S., Silva, R.M., Pinto, R.C., Salles, R.M.: Botnets: A survey. Computer Networks **57**(2), 378–403 (2013)
10. Strayer, W.T., Lapsely, D., Walsh, R., Livadas, C.: Botnet detection based on network behaviour. In: W. Lee, C. Wang, D. Dagon (eds.) Botnet Detection, Advances in Information Security, 36, 1–24. Springer (2008)
11. Unicorn race engineering: Aalborg university's formula SAE team http://unicornraceengineering.dk/ (2015)
12. Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., Garant, D.: Botnet detection based on traffic behavior analysis and flow intervals. Comput. Secur. **39**, 2–16 (2013)